Chapitre 1

Représentation de l'information

❑ Architecture d'un système ?

➢ Représente l'organisation des différentes unités/ composants d'un système et leurs interconnexions

❑ Un ordinateur?

➢ Une machine de traitement de l'information, capable de:

- Recevoir une information à travers un périphérique d'entrée.
- Mémoriser une information sur une mémoire.
- Effectuer automatiquement un traitement sur une information suivant un programme.
- Délivrer une information sur un périphérique de sortie.

Besoin du codage d'information

Un **ordinateur** manipule toute sorte **d'information** :

- Numérique
- Alphabétique
- Graphique
- Audio
- Vidéo
- Programmes.
- ...

Besoin du codage d'information

- Information numérique = information binaire
  = 1 bit (l'unité de de l'information)
- Représentée par 2 signaux électriques
- Chaque signal élémentaire peut alors se trouver dans l'un de ces deux états 0 ou 1.
- Codage : Codée par « 0 » logique ou « 1 » logique
- Différents codages pour représenter une information

Historique

- Intérêt : codage des nombres dans un but de calcul
- Apparition du calcul : préhistoire (comptage avec les cailloux et les doigts)…
- Systèmes et bases de numérotation pour représenter des nombres
- Méthodes pour compter et calculer
- **Un système de numération** est un ensemble de conventions et règles qui permet de former les nombres, les dire, les écrire et calculer.
- **Exemple:** Représentation usuelle du **145 = ?**

$$= 1 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

Les systèmes de numération les plus courants :

- Le système décimal (base 10) : utilise les 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.
- Le système binaire (base 2) : utilise les 2 chiffres : 0 et 1.
- Le système octal (base 8) : utilise les 8 chiffres : 0, 1, 2, 3, 4, 5, 6 et 7.
- Le système hexadécimal (base 16) : utilise les 16 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.

- Trois notions interviennent dans un système:
  - La **base B** du système, c'est un nombre entier quelconque.
  - Les **digits** du système sont des caractères tous différents et représentent chacun un élément de la base; il y en a donc B au total.
  - **Poids** du digit selon son rang.

- **Expression d'un nombre X en base B :**

$$\mathbf{X} = a_N B^N + a_{N-1} B^{N-1} + \ldots + a_1 B^1 + a_0 B^0$$

  - $0 \leq a_i < B$, quelque soit l'indice *i* (de 0 à N).

  - On écrit X, en ignorant les puissances de B : $\mathbf{X} = (\mathbf{a_N a_{N-1} \ldots a_1 a_0})_\mathbf{B}$

- Expression d'un nombre X en base B :

$$X = a_N B^N + a_{N-1} B^{N-1} + \ldots + a_1 B^1 + a_0 B^0$$

- **Exemple:**
  - **98_{10} = ? / 101_2 = ? / 136_8 = ? / $(127,65)_8$ = ? / $(1A7,6)_{16}$ = ?**
    - **98_{10}** $= 9 \times 10^1 + 8 \times 10^0$
    - **101_2** $= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4+1 = 5_{10}$
    - **136_8** $= 1 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 = 64+24+6 = 94_{10}$
    - **$(127,65)_8$** $= 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 6 \times 8^{-1} + 5 \times 8^{-2} = (87.828125)_{10}$
    - **$(1A7,6)_{16}$** $= 1 \times 16^2 + A \times 16^1 + 7 \times 16^0 + 6 \times 16^{-1} = (423.375)_{10}$

❑ Décimal vers base B:

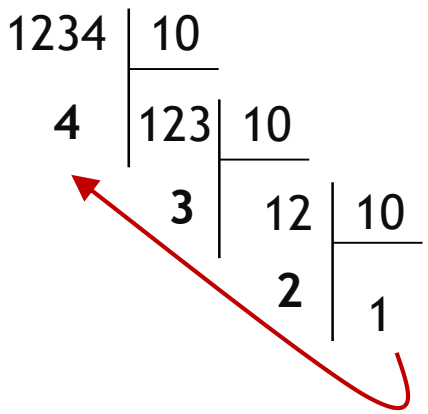
On procède par une série de divisions entières par B

- Division du nombre décimal *N* par *B* : donne une valeur v_0 et un reste r_0
- On divise v_0 par B : donne v_1 et reste r_1
- On recommence pour v_1 et ainsi de suite
- Quand $v_i < B$, c'est fini
- Le résultat de la prochaine division donnera 0
- On écrit: $(N)_B = r_i r_{i-1} ... r_1 r_0$

❑ Décimal vers base B:

❖ Exemple : $(1234)_{10}$ en décimal

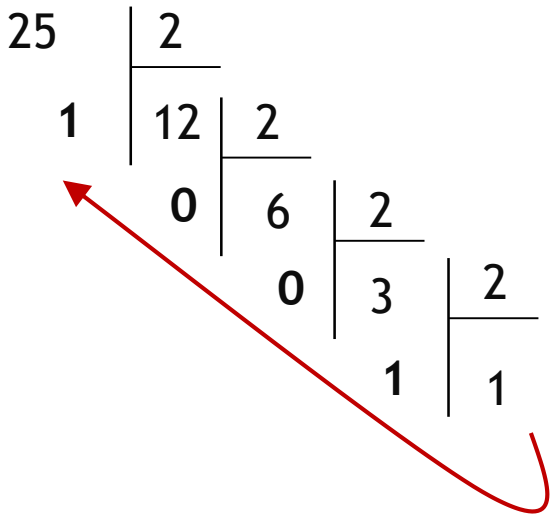
1234÷10 = 123 , reste **4**
123÷10 = 12, reste **3**
12÷10 = 1, reste **2**
1 < 10, donc on arrêt la division
On a donc : $(1234)_{10} = (1234)_{10}$



❑ Décimal vers base B:

❖ Exemple : $(25)_{10}$ en binaire

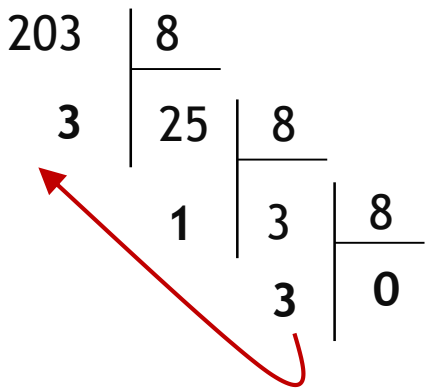
25÷2 = 12, reste **1**
12÷2 = 6, reste **0**
6÷2 = 3, reste **0**
3÷2 = 1, reste **1**
1 < 2, donc on arrêt la division
On a donc : $(25)_{10} = (11001)_2$



❑ Décimal vers base B:

❖ Exemple : $(203)_{10}$ en Octal

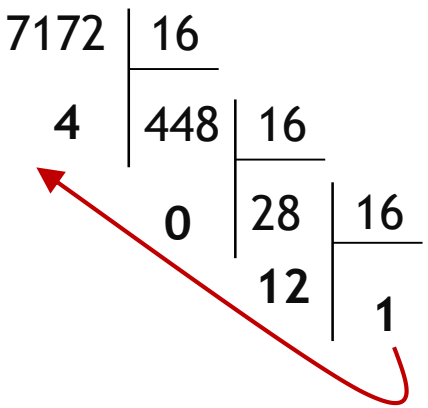
203 ÷ 8 = 25, reste **3**
25 ÷ 8 = 3, reste **1**
3 < 8, donc on arrêt la division
On a donc : $(203)_{10} = (313)_{16}$



❑ Décimal vers base B:

❖ Exemple : $(7172)_{10}$ en hexadécimal

7172 ÷ 16 = 448, reste **4**
448 ÷16 = 28, reste **0**
28 ÷16 = 1, reste **12 = C**
1 < 16, donc on arrêt la division
On a donc : $(7172)_{10} = (1C04)_{16}$



❑ Cas particuliers : Conversion du binaire à l'octal

- 1 chiffre octal = un groupe de **3 chiffres binaires**
- Avec 3 bits on code les 8 chiffres de la base octale

$(000)_2 = 0$
$(001)_2 = 1$
$(010)_2 = 2$
$(011)_2 = 3$
$(100)_2 = 4$
$(101)_2 = 5$
$(110)_2 = 6$
$(111)_2 = 7$

❑ Cas particuliers : Conversion du binaire en octal

- **Exemple 1**: $(10110001101)_2$ en octal

  - On regroupe par groupes de 3 bits : **0**10 110 001 101
  - On rajoute des zéros au début au besoin
  - $(10110001101)_2$ = $\mathbf{(2615)_8}$

- **Exemple 2:** $(1111100110)_2$ en octal

  - $(1111100110)_2 = (001\ 111\ 100\ 110)_2 = \mathbf{(1746)_8}$

❑ Cas particuliers : Conversion du l'octal en binaire

- **Exemple 1**: $(254)_8$ en binaire
  - $2 = (010)_2$, $5 = (101)_2$, $4 = (100)_2$
  - On concatène dans l'autre base ces groupes de 3 bits :

    $(254)_8 = (10101100)_2$
- **Exemple 2**: $(761)_8$ en binaire
  - $(761)_8 = (111110001)_2$
- **Exemple 3**: (831) en binaire
  - (831) ce n'est pas un nombre en octal!

❑ Cas particuliers : Conversion du binaire en hexadécimal

- 1 chiffre hexadécimal = un groupe de **4 chiffres binaires**

$(0000)_2 = 0$
$(0001)_2 = 1$
$(0010)_2 = 2$
$(0011)_2 = 3$
$(0100)_2 = 4$
$(0101)_2 = 5$
$(0110)_2 = 6$
$(0111)_2 = 7$

$(1000)_2 = 8$
$(1001)_2 = 9$
$(1010)_2 = 10 = (A)$
$(1011)_2 = 11 = (B)$
$(1100)_2 = 12 = (C)$
$(1101)_2 = 13 = (D)$
$(1110)_2 = 14 = (E)$
$(1111)_2 = 15 = (F)$

❑ Cas particuliers : Conversion du binaire en hexadécimal

- **Exemple 1**: $(10110001101)_2$ en hexadécimal
  - On regroupe par groupes de 4 bits : **0**101 1000 1101
  - $(10110001101)_2 = (58D)_{16}$

- **Exemple 2** : $(111111100111)_2$ en hexadécimal
  - $(111111100111)_2 = (1FE7)_{16}$

❑ Cas particuliers : Conversion du hexadécimal en binaire

- **Exemple 1**: $(D46C)_{16}$ en binaire
  - D = 13 = $(1101)_2$, 4 = $(0100)_2$, 6 = $(0110)_2$, C = 12 = $(1100)_2$
  - On concatène dans l'autre base ces groupes de 4 bits :

    $(D46C)_{16} = (1101010001101100)_2$

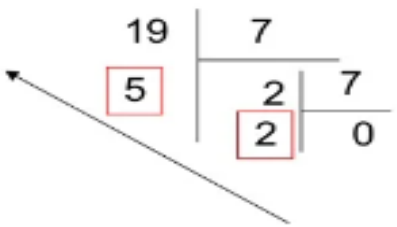
- **Exemple 2**: $(EF2A)_{16}$ en binaire
  - $(EF2A)_{16} = (1110111100101010)_2$

❑ Conversion d'une base p à une base q

- Pour convertir un nombre N écrit en base **p**, vers une base **q**, on utilise la base intermédiaire **10**.
- Mais si p et q s'écrivent respectivement sous la forme d'une puissance de 2 (4, 8 ,16,32,….), on peut passer par **la base 2** (plus rapide)

▪ **Exemple** : $(34)_5 = (?)_7$

$$(34)_5 = 3*5^1 + 4*5^0 = 15 + 4 = (19)_{10} = (?)_7$$



$$(19)_{10} = (25)_7$$
$$(34)_5 = (25)_7$$

Opérations arithmétiques

- Les opérations arithmétiques (addition, soustraction, multiplication, division) sont réalisables dans toute base B.
  - Avec mêmes règles que pour la base décimale
  - Retenues également mais dépendant de la base
    - Quand on additionne 2 chiffres a et b dans la base B
    - Si la somme des valeurs décimales de a et b dépasse ou égale B alors il y a une retenue

- **Principes de l'addition binaire**
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 1 = 10 soit 0 avec une retenue de 1

❑ Addition binaire

Exemple 1: 10 + 1011

```
   1
  0010        2
+ 1011       11
------------------
  1101       13
```

Exemple 2: 1101 + 1010

```
   1
  1101       13
+ 1010       10
--------------------
 10111       23
```

Addition de 2 nombres de 4 bits :

- on a besoin dans cet exemple de 5 bits
- Potentiel problème de débordement

❑ Addition binaire

Exemple:

```
      11                          11111
   00011010  26                00010011   19
+  00001100  12             +  00111110   62
----------------            ---------------
   00100110  38                01010001   81
```

Opérations arithmétiques

- **Principes de la soustraction binaire**
  - $0 - 0 = 0$
  - $0 - 1 = 1$ et on prend 1 à gauche
  - $1 - 0 = 1$
  - $1 - 1 = 0$

Exemple:

```
    00100101  37
-   00010001  17
      1
------------------
    00010100  20
```

Opérations arithmétiques

- **Principes de la multiplication binaire**
  - 0 × 0 = 0
  - 0 × 1 = 0
  - 1 × 0 = 0
  - 1 × 1 = 1

Exemple: 41 x 6

```
    0 0 1 0 1 0 0 1
x   0 0 0 0 0 1 1 0
-------------------
    0 0 0 0 0 0 0 0
  0 0 1 0 1 0 0 1
0 0 1 0 1 0 0 1
-----------------------------
0 0 1 1 1 1 0 1 1 0   246
```

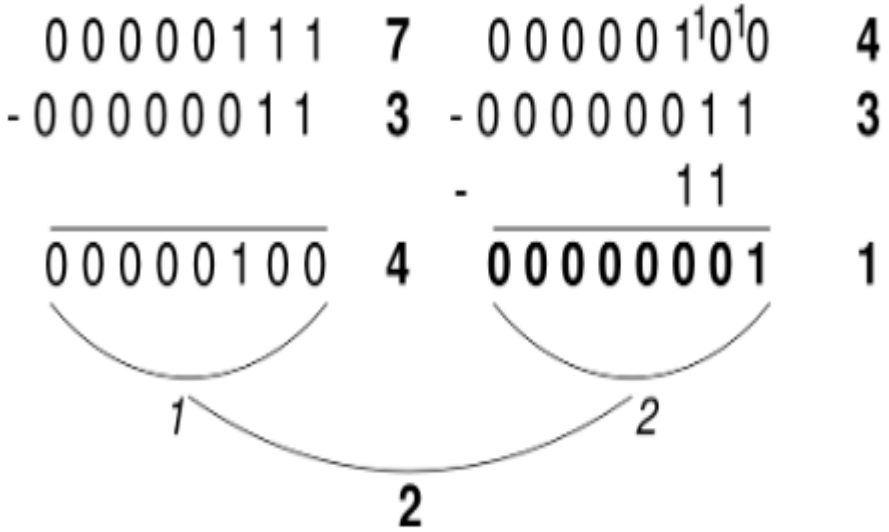
❑ Principes de la division binaire

- ➢ Division obtenue par itération de soustractions jusqu'à ce que le résultat de la soustraction soit inférieur au diviseur :
  - Quotient = nombre de soustractions
  - Reste = résultat de la dernière soustraction

- ➢ On peut aussi faire comme une division classique en décimal

- **Exemple** : 7/3

- Résultat : quotient = 2 et reste = 1



❑ Principes de la division binaire

- **Exemple : 37/5**

```
 100101 | 101
-000    | 0111
  100
- 000
  1001
- 101
  1000
- 101
  0111
- 101
  010
```

11001	101

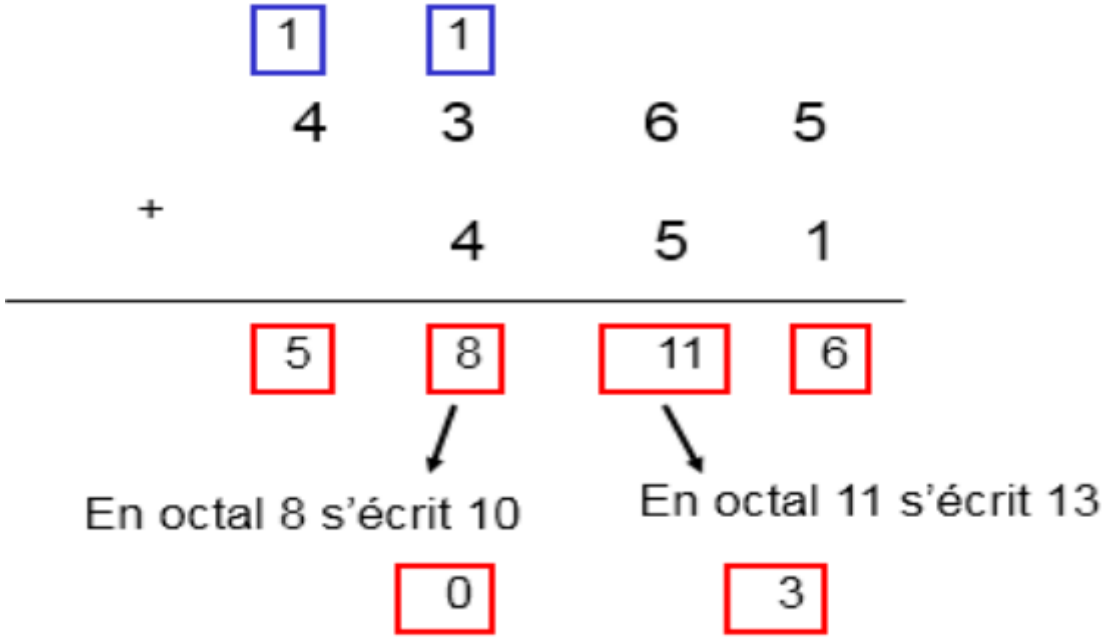
1001	11

±	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

❑ Addition en octale

- **Exemple :** $(4365)_8 + (451)_8 = ?$

$= (5036)_8$



❑ Soustraction en octale

- **Exemple :** $(347)_8 - (271)_8 = \ ?$

$= (056)_8$

	8	
	347_8	Emprunter un 8
-	$\underline{{}_1271_8}$	8+4=12
	056	12-7=5

❑ Multiplication en octale

- **Exemple :** $(24)_8 \times (7)_8 = ?$

$= (214)_8$

```
  3
   2 4
x    7
---------------------
  2 1 4
```

7*4 = 28 = $(34)_8$

14+3 = 17 = $(21)_8$

❑ Division en octale

- **Exemple :** $(651)_8 / (3)_8 = ?$

$= (215)_8 \ R = 2$

$5 \times 3 = 15 = (17)_8$

```
 651 | 3
-6   | 215
 05
 -3
  8
  21
 -17
  02
```

X /:	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Exercices d'application

- En octale

721	676	516	414	567
+604	+666	+401	+253	+337

±	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

❑ Addition en Hexadécimal

- **Exemple** : $A803_{16}+2D35_{16}$

```
  (1)
   A 8 0 3
+  2 D 3 5
  -------
   D 5 3 8
```

```
     D 2
+  5 2 9
-------
   5 F B
```

```
     1¹ F A
+  7 3 4
-------
   9 2 E
```

```
 1   5 6¹ 5
+  C 2   E
-------
 1 1 9 3
```

❑ Soustraction en Hexadécimal

- **Exemple** : $A803_{16} - 2D35_{16}$

```
   A  8  0  3
-  2  D  3  5
  (1)(1)(1)
  ----------
   7  A  C  E
```

±	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

❑ Multiplication en Hexadécimal

- **Exemple** : - $(A10)_{16} \times (5)_{16}$
  - $(C1DF)_{16} \times (8)_{16}$

```
   3
     A 1 0
       × 5
   ---------
   3 2 5 0
```

```
   6   6 7
     C 1 D F
         × 8
   -----------
   6 0 E F 8
```

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1F	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5C	63	6A
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6C	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

❑ Division en Hexadécimal

- **Exemple** : $1EC7F4_{16}$ / $7F_{16}$

```
1EC7F4 | 7F
_17D     ----
  6F7    3E0C
 _6F2
    5F4
   _5F4
      0
```

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1F	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5C	63	6A
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6C	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

$CA9EC_{16} \mid 5C_{16}$ ——— $233D_{16}$

$185850_{16} \mid 56_{16}$ ——— 4878_{16}

- **Entiers non-signés : ensemble d'entiers positifs**
- **Entiers signés: ensemble d'entiers positifs et négatifs**

Comment représenter des entiers négatifs ?

- Convention de recodage des chaînes de bits
  - Valeur signée
  - Complément à 1
  - Complément à 2

❑ Représentation par valeur signée

- Réserver un bit pour le signe (le bit le plus à gauche appelé le bit de poids fort), les autres bits codent la valeur absolue du nombre.

  - 0 = **positif** et 1 = **négatif**

- **Exemple** (sur 4 bits) :
  - $+3_{10}$= **0**011_2
  - -3_{10}= **1**011_2

- **Exemple** (sur 8 bits) :
  - $+25_{10}$ = **0**00011001 2
  - -25_{10} = **1**0011001 2

❑ Représentation par valeur signée

- Réserver un bit pour le signe (le bit le plus à gauche appelé le bit de poids fort), les autres bits codent la valeur absolue du nombre.

- **Difficultés**: Deux représentations de la valeur zéro
  - $+ 0_{10} = \mathbf{0}0000000_2$
  - $- 0_{10} = \mathbf{1}0000000_2$

- Sur 8 bits : -127…….+127

chaîne de bits	non signé	valeur signée
1111	15	−7
1110	14	−6
1101	13	−5
1100	12	−4
1011	11	−3
1010	10	−2
1001	9	−1
1000	8	−0
0111	7	7
0110	6	6
0101	5	5
0100	4	4
0011	3	3
0010	2	2
0001	1	1
0000	0	0

❑ Complément à 1

- Le bit de poids fort correspond au signe :
- 0 = **positif** 1 = **négatif**

- Un nombre négatif s'obtient en complémentant bit à bit sa valeur absolue avec 1. Ou on prend la représentation de la partie entière et on inverse tous les bits.

- **Exemple**: représentation de -4_{10} sur 4 bits
  $4_{10} = 0100_2$ donc $-4_{10} = 1011_2$

```
  1111
- 0100
------
  1011
```

- **Exemple**: représentation de -25_{10} sur 8 bits :
  $25_{10} = 00011001_2$ donc -25_{10} :11100110_2

```
  11111111
- 00011001
----------
  11100110
```

❑ Complément à 1

- Le bit de poids fort correspond au signe :
- 0 = **positif** 1 = **négatif**

- Deux représentations pour 0 :
  00000000$_2$ et **1**1111111$_2$

- Nombres représentés sur 8 bits : -127……+127
- Valeur max (complément à 1):
  - **Positive** de 00000000$_2$ à 01111111$_2$ => de **0** à **127$_{10}$**
  - **Négative** de 10000000$_2$ à 11111111$_2$ => de **-127** à **-0$_{10}$**

chaîne de bits	non signé	complément à 1
1111	15	−0
1110	14	−1
1101	13	−2
1100	12	−3
1011	11	−4
1010	10	−5
1001	9	−6
1000	8	−7
0111	7	7
0110	6	6
0101	5	5
0100	4	4
0011	3	3
0010	2	2
0001	1	1
0000	0	0

❑ Complément à 2

- Le bit de poids fort correspond au signe :
- 0 = **positif** 1 = **négatif**

- Un nombre négatif s'obtient en ajoutant 1 au complément à 1 de sa valeur absolue. Ou on prend la représentation de la partie entière et on soustrait 1 puis on inverse tous les bits.

- **Exemple (sur 4 bits):** représentation de -6_{10}
  - $+ 6_{10} = \mathbf{0110_2}$
  - Complément à 1 de $+6_{10} = \mathbf{1001_2}$
  - Ajout de 1: $1001_2 + 1 = \mathbf{1010_2}$
  - $-6_{10} = \mathbf{1010_2}$

❑ Complément à 2

- Le bit de poids fort correspond au signe :
- 0 = **positif** 1 = **négatif**

- Un nombre négatif s'obtient en ajoutant 1 au complément à 1 de sa valeur absolue. Ou on prend la représentation de la partie entière et on soustrait 1 puis on inverse tous les bits.

- **Exemple** (sur 8 bits): représentation de -25_{10}
  - $+25_{10} = 00011001_2$
  - Complément à 1 de $+25_{10} = 11100110_2$
  - Ajout de 1: $11100111_2 + 1 = 11100111_2$
  - $-25_{10} = 11100111_2$

❑ Complément à 2

- Le bit de poids fort correspond au signe :
- 0 = **positif** 1 = **négatif**

- Une seule représentation pour 0 : 00000000_2

- Nombres représentés sur 8 bits : -128..+127

- Valeur max (complément à 2):
  - **Positive** de 00000000_2 à 01111111_2 => de **0** à **127_{10}**
  - **Négative** de 10000000_2 à 11111111_2 => de **-128** à **-1_{10}**

chaîne de bits	non signé	complément à 2
1111	15	−1
1110	14	−2
1101	13	−3
1100	12	−4
1011	11	−5
1010	10	−6
1001	9	−7
1000	8	−8
0111	7	7
0110	6	6
0101	5	5
0100	4	4
0011	3	3
0010	2	2
0001	1	1
0000	0	0

Exercice:

- Convertir en décimal:
  - (11001000)CA2
  - (00100111)CA2

Les systèmes de numération: Les entiers signés

- **Capacité de représentation**
  - Les valeurs positives: intervalle [0, $2^{\text{nombre bits}}$ -1]
  - Les valeurs négative (complément à 1): intervalle [1-($2^{\text{nombre bits}}$)/2, -1+($2^{\text{nombre bits}}$)/2]
  - Pour négative (complément à 2): intervalle [-($2^{\text{nombre bits}}$)/2, -1+($2^{\text{nombre bits}}$)/2]
- **Exemple pour 3 bits:**
  - Valeurs positifs: intervalle [0, 2^3 -1]=>[0, 7]
  - Valeurs négatives (complément à 1): intervalle [1-(2^3)/2, -1+(2^3)/2]

    =>[1-4, -1+4] = [-3,3]
  - Valeurs négatives (complément à 2): intervalle [-(2^3)/2, -1+(2^3)/2]

    =>[-4, -1+4] = [-4,3]

- Application: Compléter le tableau suivant !

	Décimal		
	Valeur signée	Complément à 1	Complément à 2
011			
010			
001			
000			
111			
110			
101			
100			

Les systèmes de numération: Les entiers signés

- Application:

	Valeur signée	Complément à 1	Complément à 2
011	3	3	3
010	2	2	2
001	1	1	1
000	0	0	0
111	-3	-0	-1
110	-2	-1	-2
101	-1	-2	-3
100	-0	-3	-4

Les systèmes de numération: Les entiers signés

Binaire	Décimal	Valeur signée	Complément à 1	Complément à 2
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

Retenu & Débordement

- **Retenu** est rencontrée lorsqu'une retenue existe à la fin de l'opération d'addition ou soustraction(comme décimal).

- **Débordement** : la taille allouée (8, 16 ... bits) au codage d'un nombre est trop petite pour coder ou stocker correctement le résultat d'un calcul.

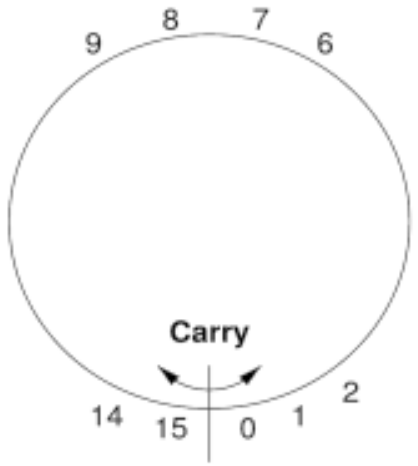
- Exemple avec addition, sur 8 bits, non signé :
  - **1011 0011 + 1000 0101 = 100111000**
  - Besoin de 9 bits pour coder le nombre
  - ➔ Stockage du résultat impossible sur 8 bits

- Exemple avec addition, sur 8 bits, signé :
  - **01110011 + 01000101 = 10111000**
  - Addition de 2 positifs donne un négatif !

Retenu & Débordement

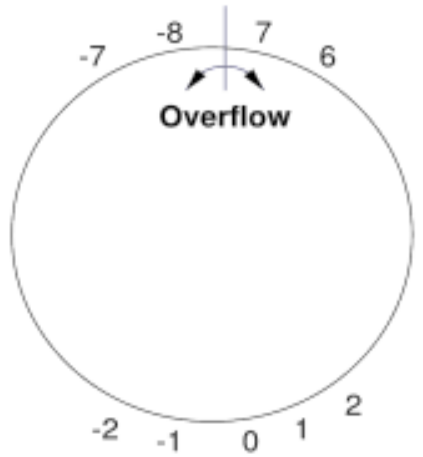
- **Exemple 1 :(4bits** signé **)**
  - 7 + 3 = ?

0 1 1 1
\+ **0** 0 1 1
---
1 0 1 0 -6 !!!



Non signé		Signé	
1110	14	1110	-2
+ 1011	11	+ 1011	-5
11001	25>15	11001	-7

Détection :
1 bit de plus en sortie



Non signé		Signé	
0111	7	0111	7
+ 0001	1	+ 0001	1
1000	8	1000	-8

Détection :
Opérandes de même signe
Résultat de signe différent

- **Exemple 2:(8bits** signé **)**
  - 110 + 43 = ?

0 1 1 0 1 1 1 0 110
\+ **0** 0 1 0 1 0 1 1 43
---
1 0 0 1 1 0 0 1 (-103) !! (et non 153)

- Méthode de la virgule fixe
- Méthode de la virgule flottante

❑ Méthode de la virgule fixe

- La position de la virgule reste inchangée.
- On code un nombre réel R sur 3 champs :
  - Le bit de poids fort (MSB) est réservé pour le signe de R.
  - La partie entière de R est codé en binaire (sur un nombre de bits précis).
  - La partie fractionnaire de R est également codé en binaire (sur un nombre de bits précis).

Bit de signe	Partie entière en binaire	Partie fractionnaire en binaire

❑ Méthode de la virgule fixe

- **Exemple 1:**
  - R = -10.25
  - Signe = - donc MSB = **1**
  - Partie entière de R = 10 = $(\mathbf{1010})_2$
  - Partie fractionnaire de R = 0.25 = $(\mathbf{01})_2$
    - 0,25 x 2 = **0**.5
    - 0,5 x 2 = **1**
  - La représentation complète de R sur 2 octets (1 octet pour la partie entière plus le signe et 1 octet pour la partie fractionnaire):
  - R = **1**0001010**01000000**

❑ Méthode de la virgule fixe

- **Exemple :** conversion de 14.375_{10} en base 2 **?**
  - $14_{10} = 1110_2$
  - $0.375_{10} = (?)_2$
    - $0.375 \times 2 = \mathbf{0}.75$
    - $\mathbf{0}.75 \times 2 = \mathbf{1}.5$
    - $\mathbf{0}.5 \times 2 = \mathbf{1}.0$
  - Résultat = $14.375_{10} = 1110.011_2$

- **Exemple de conversion binaire en virgule fixe :**
  - $110.011_2 = ?$
  - $1x\mathbf{2^2}+1x\mathbf{2^1}+0x\mathbf{2^0}+0x\mathbf{2^{-1}}+1x\mathbf{2^{-2}}+1x\mathbf{2^{-3}} = 6.375_{10}$

- **Méthode de la virgule fixe**
  - **Exercice:**
  - Trouver la représentation binaire de 0.2357 sur 8 bits?
  - $0.2357 \times 2 = 0 + 0.4714$
  - $0.4714 \times 2 = 0 + 0.9428$
  - $0.9428 \times 2 = 1 + 0.8856$
  - $0.8856 \times 2 = 1 + 0.7712$
  - $0.7712 \times 2 = 1 + 0.5424$
  - $0.5424 \times 2 = 1 + 0.0848$
  - $0.0848 \times 2 = 0 + 0.1696$
  - $0.1696 \times 2 = 0 + 0.3392$
  - $(0.2357)10 = (0.00111100)_2$

❑ Méthode de la virgule flottante

- La position de la virgule est variable.
- On écrit le nombre réel sous la forme :

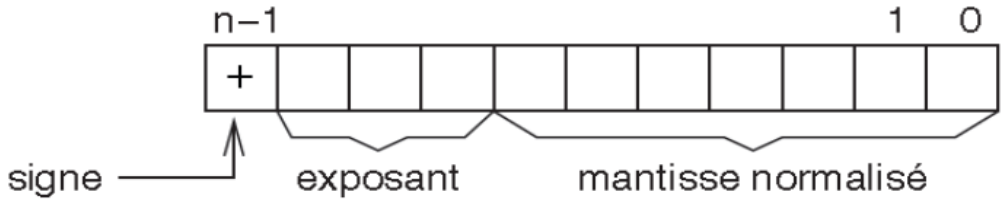
$$R = (\pm 1) \times M \times 2^{E}$$

  - un bit de signe
  - une mantisse M
  - un exposant E

- Exemple de représentation
  - $0.0000111010 \cdot 2^{0}$
  - $0.000000111010 \cdot 2^{2}$
  - $1.111010 \cdot 2^{-5}$

❑ Méthode de la virgule flottante

- Codage IEEE 754
  - Le signe + est représenté par 0 et le signe − par 1
  - La mantisse appartient à l'intervalle [1, 2[
  - L'exposant est un entier relatif et il est établi de manière à ce que la mantisse soit de la forme « 1,… »



- **Plusieurs formats:**
  - Simple précision : **32** bits (soit 4 octets) 1 bit de signe, 8 bits d'exposant, 23 bits de mantisse
  - Double précision : **64** bits (soit 8 octets) 1 bit de signe, 11 bits d'exposant, 52 bits de mantisse

❑ Méthode de la virgule flottante

- Codage IEEE 754 (**simple précision**)
  - Au lieu de coder **E** (en binaire), on code **E + 127.**
  - C'est l'exposant biaisé $\mathbf{E_b}$: $\mathbf{E_b = E + 127}$
  - Cela permet d'avoir un **exposant non signé**

❑ Méthode de la virgule flottante

- Codage IEEE 754 (**simple précision 32** bits)
- Application:
  - R = −6.625
  - $6.625_{10} = 110.101_2$
  - $110.1010_2 = 1.\mathbf{10101} \times 2^2$
  - M = **10101**00000000000000000
  - E= 2 => $E_b = 2+127 = 129_{10} = 10000001_2$
  - Résultat : **1** 10000001 **10101**00000000000000000

❑ Méthode de la virgule flottante

- Codage IEEE 754 (**simple précision 32** bits)
- Application:
  - $R = -0.75$
  - $0.75_{10} = 0.11_2$
  - $0.11_2 = 1.\mathbf{1} \times 2^{-1}$
  - M = **1**00000000000000000000
  - E= -1 => E_b = -1+127 = $126_{10} = 1111110_2$
  - Résultat : **1** 01111110 **1**00000000000000000000

❑ Définition

- Codage : Opération consistant à représenter des informations à l'aide d'un code.
- Les codes sont des combinaisons de bits permettant de représenter sous forme binaire des informations quelconques.

❖ **Le Code binaire pur ou code binaire naturel**

❖ **Le code BCD (décimal codé binaire)**

❖ **Le code gray ou code binaire reflechi**

❖ **Le Code ASCII**

❖ Le code binaire pur ou code binaire naturel

- Le code binaire naturel est utilisé pour représenter un nombre dans la base de numération binaire.
- Pour une représentation sur 4 bit, les nombres sont codés par:

CODE BINAIRE	EQUIVALENT DECIMAL
0000	0
0001	1
0010	2
:	:
1110	14
1111	15

❖ Le code BCD (décimal codé binaire)

- Un digit décimal peut atteindre la valeur 9, il faut donc nécessairement 4 bits pour coder chaque digit.
- Il suffit de transformer chaque chiffre en binaire naturel sur 4 bits, sans faire de calcul.
- **Exemple :**
  - (1995)**10** = (0001 1001 1001 0101)**BCD**
  - (874)**10** = (1000 0111 0100)**BCD**
  - (1001 0011)**BCD** = (93)**10**

✓ **RQ: BCD # binaire pur**

- (137)**10** = (10001001)**2** = (0001 0011 0111)**BCD**

❖ Le code BCD (décimal codé binaire)

❑ L'addition des nombres en BCD

- Il y a deux cas possible:
  - la somme est un chiffre BCD valide (0 à 9); ou,
  - la somme est égale ou supérieure à dix.

• Dans le premier cas, aucune action n'est requise.

• Dans le deuxième cas, il faut ajouter +6 à la somme obtenue. La combinaison de la retenue et de la somme finale produit alors un résultat correct.

❖ Le code BCD (décimal codé binaire)

❑ L'addition des nombres en BCD

- Exemple : addition 3 + 4 en BCD :

```
   3        0 0 1 1
 + 4      + 0 1 0 0
  07     0 | 0 1 1 1   ← somme inférieure à dix, résultat correct
                        (retenue 0, somme 7)
```

- Exemple : addition 7 + 8 en BCD :

```
   7        0 1 1 1
 + 8      + 1 0 0 0
  15     0 | 1 1 1 1   ← somme supérieure à 9, il faut ajouter +6
         + 0 1 1 0     + 6
        1 | 0 1 0 1   ← résultat correct, retenue de 1, somme de 5
```

- Exemple : addition 9 + 8 en BCD :

```
   9        1 0 0 1
 + 8      + 1 0 0 0
  17     1 | 0 0 0 1   ← somme supérieure à 9, il faut ajouter +6
         + 0 1 1 0     + 6
        1 | 0 1 1 1   ← résultat correct, retenue de 1, somme de 7
```

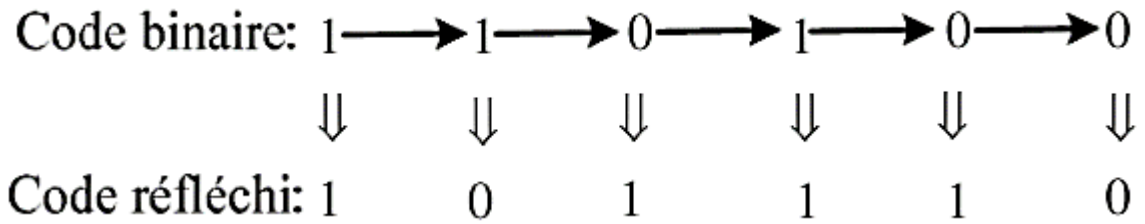
❖ Le code gray ou code binaire reflechi

❖ C'est un code de même densité que le code binaire pur (avec n bit, on code 2n Nombre)

- On passe d'un nombre à son suivant en modifiant un seul bit.

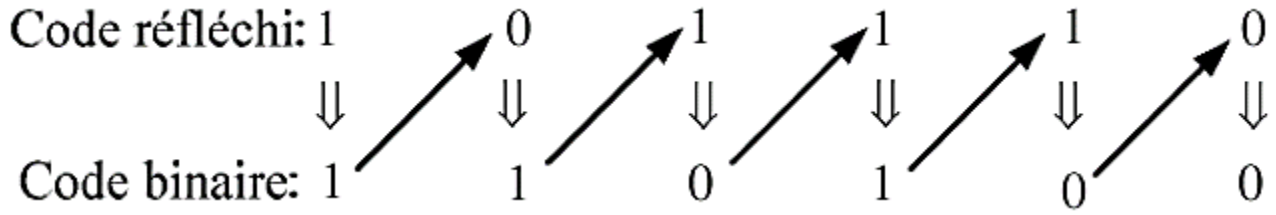
Binaire Gray	Décimal
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9

- **Conversion binaire en code réfléchi (Gray):**
  - Ecrire le nombre binaire à convertir.
  - Reproduire le bit binaire de plus fort bit, pour obtenir celui du code réfléchi.
  - Le reste des bits du code de réfléchi sont obtenus en additionnant binaires deux à deux, à partir de la gauche, sans tenir compte de la retenue.
- **Exemple: 110100 = ? gray**



❖ Le code gray ou code binaire reflechi

- **Code réfléchi (Gray) en code binaire :**
  - Ecrire le nombre binaire à convertir.
  - Reproduire le bit de plus fort poids du nombre réfléchi, pour obtenir celui du code de binaire
  - Le reste des bits du code binaire sont obtenus en additionnant les bits sans retenue, deux à deux, à partir de la gauche, le bit du rang i du code binaire avec celui du rang i-1, est ainsi de suite.
- **Exemple: (101110)g en binaire**



Systèmes de codage - Représentation des caractères

- La mémoire de l'ordinateur conserve toutes les données sous forme numérique. Il n'existe pas de méthode pour stocker directement les caractères.

- Plusieurs formats pour représenter des caractères (symboles alphanumériques) sous forme binaire :

  - **ASCII** (American Standard Code for Information Interchange)
    - Représentation sur 7 bits (pas d'accents)
    - ASCII étendu : sur 8 bits mais pas de normalisation

  - **Unicode** : encodage sur 16 bits (65536 possibilités) pour représenter tous les caractères de toutes les langues (Arabic, Basic Latin…)

Représentation des caractères

❑ Exemple: Table code ASCII

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Le code de U (majuscule) est représenté par:

- 85 en code décimale
- 55 en code hexadécimal
- u (minuscule) est $1110101_{(2)}$, soit $117_{(10)}$ soit $75_{(16)}$.
- Le code ascii $1000001 = 41_{(16)} = 65_{(10)}$.

❑Programmes de conversion

- Un programme permettant de convertir un nombre d'une base A vers une base B (A et B compris entre 2 et 16).

- Un programme permettant de représenter les nombres négatifs en utilisant les 3 méthodes : VS, Cà1, Cà2
- +Tester des cas (retenu et débordement)

- Un programme permettant de représenter les nombres à virgule fixe et virgule flottante.